1. Kudu官网

Kudu官网：https://kudu.apache.org/

2. 编译

RHEL or CentOS 6.6 or later is required to build Kudu from source. To build on a version older than 7.0, the Red Hat Developer Toolset must be installed (in order to have access to a C++11 capable compiler).

2.1. 第一步：安装一些依赖软件

Install the prerequisite libraries, if they are not installed.

安装一些必要的库

这其中，会安装 git 用于拉取 kudu 源码，安装 openjdk，按需增删。

```
sudo yum install autoconf automake cyrus-sasl-devel cyrus-sasl-gssapi \
  cyrus-sasl-plain flex gcc gcc-c++ gdb git java-1.8.0-openjdk-devel \
  krb5-server krb5-workstation libtool make openssl-devel patch \
  pkgconfig redhat-lsb-core rsync unzip vim-common which -y
```

如果已经安装有 git 和 JDK：

```
sudo yum install autoconf automake cyrus-sasl-devel cyrus-sasl-gssapi \
  cyrus-sasl-plain flex gcc gcc-c++ gdb \
  krb5-server krb5-workstation libtool make openssl-devel patch \
  pkgconfig redhat-lsb-core rsync unzip vim-common which -y
```

2.2. 第二步：安装 Red Hat Developer Toolset

If building on RHEL or CentOS older than 7.0, install the Red Hat Developer Toolset.

安装一下 Red Hat Developer Toolset

```
DTLS_RPM=rhscl-devtoolset-3-epel-6-x86_64-1-2.noarch.rpm
DTLS_RPM_URL=https://www.softwarecollections.org/repos/rhscl/devtoolset-3/epel-6-x86_64/noarch/${DTLS_RPM}
wget ${DTLS_RPM_URL} -O ${DTLS_RPM}
sudo yum install -y scl-utils ${DTLS_RPM}
sudo yum install -y devtoolset-3-toolchain
```

2.3. 第三步：安装 NVM 功能

这一步是可选的，如果你想使用 kudu 的 NVM 功能的话。就执行以下的安装操作。

Optional: If support for Kudu's NVM (non-volatile memory) block cache is desired, install the memkind library.

```
sudo yum install memkind -y
```

If the memkind package provided with the Linux distribution is too old (1.8.0 or newer is required), build and install it from source.

```
sudo yum install numactl-libs numactl-devel -y
git clone https://github.com/memkind/memkind.git
cd memkind
./build.sh --prefix=/usr
sudo yum remove memkind -y
sudo make install
sudo ldconfig
```

2.4. 第四步：安装一些额外的依赖库

可选的，安装一些额外的依赖库，比如ruby等

Optional: Install some additional packages, including ruby, if you plan to build documentation.

```
sudo yum install doxygen gem graphviz ruby-devel zlib-devel -y
```

If building on RHEL or CentOS older than 7.0, the gem package may need to be replaced with rubygems

2.5. 第五步：下载 kudu 的源码

Clone the Git repository and change to the new `kudu` directory.

```
git clone https://github.com/apache/kudu
cd kudu
```

如果是自己从官网下载的，则解压缩一下：

```
tar -zxvf ~/soft/apache-kudu-1.11.1.tar.gz -C ~/apps/
cd /root/apps/apache-kudu-1.11.1
```

2.6. 第六步：编译依赖组件

使用 build-if-necessary.sh 脚本去编译一些缺失的第三方依赖。这一步操作，很耗费时间，我第一次操作花费了3个小时

Build any missing third-party requirements using the `build-if-necessary.sh` script. Not using the devtoolset will result in `Host compiler appears to require libatomic, but cannot find it.`

```
build-support/enable_devtoolset.sh thirdparty/build-if-necessary.sh
```

这一步的时间，相当的长！请耐心。

Built Success：

```
+ '[' -n '' -o -n '' ']'
+ '[' -n '' -o -n '' ']'
+ '[' -n '' -o -n '' ']'
+ '[' -n '' -o -n '' ']'
+ '[' -n '' -o -n '' ']'
+ restore_env
+ PREFIX=/root/apps/apache-kudu-1.11.1/thirdparty/installed/tsan
+ MODE_SUFFIX=.tsan
+ EXTRA_CFLAGS='-fno-omit-frame-pointer  '
+ EXTRA_CXXFLAGS='-fno-omit-frame-pointer -O2  '
+ EXTRA_LDFLAGS=' '
+ EXTRA_LIBS=' '
+ finish
+ /root/apps/apache-kudu-1.11.1/thirdparty/postflight.py
Running post-flight checks
------------------------
Checking that no TSAN dependency depends on libstdc++ ... PASSED
------------------------
Post-flight checks succeeded.
+ echo --------------------
--------------------
+ echo 'Thirdparty dependencies '\''common uninstrumented'\'' built and installed successfully'
Thirdparty dependencies 'common uninstrumented' built and installed successfully
+ exit 0
[root@bigdata99 apache-kudu-1.11.1]#
```

2.7. 第七步：编译

执行编译操作！需要选择一个任意临时目录作为输出。

Build Kudu, using the utilities installed in the previous step. Choose a build directory for the intermediate output, which can be anywhere in your filesystem except for the `kudu` directory itself. Notice that the devtoolset must still be specified, else you'll get `cc1plus: error: unrecognized command line option "-std=c++11"`.

```
cd ~/apps/apache-kudu-1.11.1
mkdir -p build/release
cd build/release

../../build-support/enable_devtoolset.sh \
../../thirdparty/installed/common/bin/cmake \
-DCMAKE_BUILD_TYPE=release ../..

make -j4
```

If you need to install only a subset of Kudu executables, you can set the following `cmake` flags to OFF in order to skip any of the executables.

KUDU_CLIENT_INSTALL (set to OFF to skip installing /usr/local/bin/kudu executable)
KUDU_TSERVER_INSTALL (set to OFF to skip installing /usr/local/sbin/kudu-tserver executable)
KUDU_MASTER_INSTALL (set to OFF to skip installing /usr/local/sbin/kudu-master executable)

E.g., use the following variation of `cmake` command if you need to install only Kudu client libraries and headers:

```
../../build-support/enable_devtoolset.sh \
  ../../thirdparty/installed/common/bin/cmake \
  -DKUDU_CLIENT_INSTALL=OFF \
  -DKUDU_MASTER_INSTALL=OFF \
  -DKUDU_TSERVER_INSTALL=OFF
  -DCMAKE_BUILD_TYPE=release ../..
```

```
Scanning dependencies of target ksck_remote-test
[ 98%] Built target ksck-test
Scanning dependencies of target tpch_real_world
[ 98%] Building CXX object src/kudu/tools/CMakeFiles/ksck_remote-test.dir/ksck_remote-test.cc.o
[ 98%] Building CXX object src/kudu/benchmarks/CMakeFiles/tpch_real_world.dir/tpch/tpch_real_world.cc.o
[ 98%] Building CXX object src/kudu/tools/CMakeFiles/kudu-tool-test.dir/kudu-tool-test.cc.o
[ 98%] Built target kudu-admin-test
Scanning dependencies of target tpch1
[ 98%] Building CXX object src/kudu/benchmarks/CMakeFiles/tpch1.dir/tpch/tpch1.cc.o
[ 98%] Linking CXX executable ../../../bin/tpch_real_world
[ 98%] Built target tpch_real_world
Scanning dependencies of target rpc_line_item_dao-test
[ 98%] Building CXX object src/kudu/benchmarks/CMakeFiles/rpc_line_item_dao-test.dir/tpch/rpc_line_item_dao-test.cc.o
[ 98%] Linking CXX executable ../../../bin/tpch1
[ 98%] Linking CXX executable ../../../bin/ksck_remote-test
[ 98%] Linking CXX executable ../../../bin/rpc_line_item_dao-test
[ 98%] Built target tpch1
[ 98%] Built target rpc_line_item_dao-test
[ 98%] Built target ksck_remote-test
[100%] Linking CXX executable ../../../bin/kudu-tool-test
[100%] Built target kudu-tool-test
[root@bigdata99 release]#
```

2.8. 第八步：安装 kudu

Optional: install Kudu executables, libraries and headers.

Running `sudo make install` installs the following:

```
kudu-tserver and kudu-master executables in /usr/local/sbin
Kudu command line tool in /usr/local/bin
Kudu client library in /usr/local/lib64/
Kudu client headers in /usr/local/include/kudu
```

The default installation directory is `/usr/local`. You can customize it through the `DESTDIR` environment variable.

```
sudo make DESTDIR=/root/apps/kudu-1.11.1 install
```

日志：

```
[root@bigdata99 release]# sudo make DESTDIR=/root/apps/kudu-1.11.1 install
Install the project...
-- Install configuration: "RELEASE"
-- Installing: /root/apps/kudu-1.11.1/usr/local/lib64/libkudu_client.so.0.1.0
-- Installing: /root/apps/kudu-1.11.1/usr/local/lib64/libkudu_client.so.0
-- Installing: /root/apps/kudu-1.11.1/usr/local/lib64/libkudu_client.so
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/client/callbacks.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/client/client.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/client/row_result.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/client/scan_batch.h
-- Installing: /root/apps/kudu-
1.11.1/usr/local/include/kudu/client/scan_predicate.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/client/schema.h
```

```
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/client/shared_ptr.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/client/stubs.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/client/value.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/client/write_op.h
-- Installing: /root/apps/kudu-
1.11.1/usr/local/include/kudu/client/resource_metrics.h
-- Installing: /root/apps/kudu-
1.11.1/usr/local/include/kudu/common/partial_row.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/util/kudu_export.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/util/int128.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/util/monotime.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/util/slice.h
-- Installing: /root/apps/kudu-1.11.1/usr/local/include/kudu/util/status.h
-- Installing: /root/apps/kudu-
1.11.1/usr/local/share/doc/kuduClient/examples/CMakeLists.txt
-- Installing: /root/apps/kudu-
1.11.1/usr/local/share/doc/kuduClient/examples/example.cc
-- Installing: /root/apps/kudu-
1.11.1/usr/local/share/kuduClient/cmake/kuduClientTargets.cmake
-- Installing: /root/apps/kudu-
1.11.1/usr/local/share/kuduClient/cmake/kuduClientTargets-release.cmake
-- Installing: /root/apps/kudu-
1.11.1/usr/local/share/kuduClient/cmake/kuduClientConfig.cmake
-- Munging kudu client targets in /root/apps/kudu-
1.11.1/usr/local/share/kuduClient/cmake/kuduClientConfig.cmake
-- Munging kudu client targets in /root/apps/kudu-
1.11.1/usr/local/share/kuduClient/cmake/kuduClientTargets-release.cmake
-- Munging kudu client targets in /root/apps/kudu-
1.11.1/usr/local/share/kuduClient/cmake/kuduClientTargets.cmake
-- Installing: /root/apps/kudu-1.11.1/usr/local/sbin/kudu-master
-- Installing: /root/apps/kudu-1.11.1/usr/local/bin/kudu
-- Installing: /root/apps/kudu-1.11.1/usr/local/sbin/kudu-tserver
```

```
[root@bigdata99 local]# cd /root/apps/kudu-1.11.1/usr/local
[root@bigdata99 local]# ll
total 0
drwxr-xr-x. 2 root root 18 Jul 28 17:23 bin
drwxr-xr-x. 3 root root 18 Jul 28 17:23 include
drwxr-xr-x. 2 root root 89 Jul 28 17:23 lib64
drwxr-xr-x. 2 root root 45 Jul 28 17:23 sbin
drwxr-xr-x. 4 root root 35 Jul 28 17:23 share
[root@bigdata99 local]#
```

2.9. 第九步：编译文档

可选的，编译文档!

Optional: Build the documentation. NOTE: This command builds local documentation that is not appropriate for uploading to the Kudu website.

```
make docs
```

2.10. 第十步：处理 lib

对 lib 做软连接

```
ln -s /root/apps/kudu-1.11.1/usr/local/include/* /usr/local/include/
ln -s /root/apps/kudu-1.11.1/usr/local/lib64/* /usr/local/lib64/
ln -s /root/apps/kudu-1.11.1/usr/local/share/* /usr/local/share/
```

2.11. 第十一步：修改配置

创建 conf 目录，创建两个配置文件：

```
mkdir conf
cd conf
touch master.gflagfile
touch tserver.gflagfile
```

配置文件的具体配置信息为：

master.gflagfile

```
## Comma-separated list of the RPC addresses belonging to all Masters in this
cluster.
## NOTE: if not specified, configures a non-replicated Master.
--master_addresses=bigdata02:7051,bigdata03:7051,bigdata04:7051
--rpc_bind_addresses=bigdata02:7051

--log_dir=/data/kudu_data/master/logs
--log_filename=kudu2
--fs_wal_dir=/data/kudu_data/master/wal
--fs_data_dirs=/data/kudu_data/master/data
--enable_process_lifetime_heap_profiling=true
--heap_profile_path=/data/kudu_data/master/heap

--rpc-encryption=disabled
--rpc_authentication=disabled
#--unlock_unsafe_flags=true
#--allow_unsafe_replication_factor=true

#--max_log_size=1800
--max_log_size=2048

#--memory_limit_hard_bytes=0
--memory_limit_hard_bytes=1073741824

--default_num_replicas=3
--max_clock_sync_error_usec=10000000
--consensus_rpc_timeout_ms=30000
--follower_unavailable_considered_failed_sec=300
--leader_failure_max_missed_heartbeat_periods=3
#--block_manager_max_open_files=10240
#--server_thread_pool_max_thread_count=-1
--tserver_unresponsive_timeout_ms=60000
```

```
--rpc_num_service_threads=10
--max_negotiation_threads=50
--min_negotiation_threads=10
--rpc_negotiation_timeout_ms=3000
--rpc_default_keepalive_time_ms=65000

#--rpc_num_acceptors_per_address=1
--rpc_num_acceptors_per_address=5

#--master_ts_rpc_timeout_ms=30000
--master_ts_rpc_timeout_ms=60000

#--remember_clients_ttl_ms=60000
--remember_clients_ttl_ms=3600000

#--remember_responses_ttl_ms=60000
--remember_responses_ttl_ms=600000

#--rpc_service_queue_length=50
--rpc_service_queue_length=1000

#--raft_heartbeat_interval_ms=500
--raft_heartbeat_interval_ms=60000

#--heartbeat_interval_ms=1000
--heartbeat_interval_ms=60000
--heartbeat_max_failures_before_backoff=3

## You can avoid the dependency on ntpd by running Kudu with --use-hybrid-
clock=false
## This is not recommended for production environment.
## NOTE: If you run without hybrid time the tablet history GC will not work.
## Therefore when you delete or update a row the history of that data will be
kept
## forever. Eventually you may run out of disk space.
--use_hybrid_clock=false

--webserver_enabled=true
--metrics_log_interval_ms=60000
--webserver_port=8051
#--webserver_doc_root=/data/kudu/www
```

tserver.gflagfile

```
## Comma-separated list of the RPC addresses belonging to all Masters in this
cluster.
## NOTE: if not specified, configures a non-replicated Master.
--tserver_master_addrs=bigdata02:7051,bigdata03:7051,bigdata04:7051
--rpc_bind_addresses=bigdata02:7050

--log_dir=/data/kudu_data/tserver/logs
--log_filename=kudu2
--fs_wal_dir=/data/kudu_data/tserver/wal
--fs_data_dirs=/data/kudu_data/tserver/data
--enable_process_lifetime_heap_profiling=true
--heap_profile_path=/data/kudu_data/tserver/heap
```

```
--rpc-encryption=disabled
--rpc_authentication=disabled
#--unlock_unsafe_flags=true
#--allow_unsafe_replication_factor=true

#--max_log_size=1800
--max_log_size=2048

#--memory_limit_hard_bytes=0
--memory_limit_hard_bytes=1073741824

--default_num_replicas=3
--max_clock_sync_error_usec=10000000
--consensus_rpc_timeout_ms=30000
--follower_unavailable_considered_failed_sec=300
--leader_failure_max_missed_heartbeat_periods=3
#--block_manager_max_open_files=10240
#--server_thread_pool_max_thread_count=-1
--tserver_unresponsive_timeout_ms=60000
--rpc_num_service_threads=10
--max_negotiation_threads=50
--min_negotiation_threads=10
--rpc_negotiation_timeout_ms=3000
--rpc_default_keepalive_time_ms=65000

#--rpc_num_acceptors_per_address=1
--rpc_num_acceptors_per_address=5

#--master_ts_rpc_timeout_ms=30000
--master_ts_rpc_timeout_ms=60000

#--remember_clients_ttl_ms=60000
--remember_clients_ttl_ms=3600000

#--remember_responses_ttl_ms=60000
--remember_responses_ttl_ms=600000

#--rpc_service_queue_length=50
--rpc_service_queue_length=1000

#--raft_heartbeat_interval_ms=500
--raft_heartbeat_interval_ms=60000

#--heartbeat_interval_ms=1000
--heartbeat_interval_ms=60000
--heartbeat_max_failures_before_backoff=3

## You can avoid the dependency on ntpd by running Kudu with --use-hybrid-
clock=false
## This is not recommended for production environment.
## NOTE: If you run without hybrid time the tablet history GC will not work.
## Therefore when you delete or update a row the history of that data will be
kept
## forever. Eventually you may run out of disk space.
--use_hybrid_clock=false

--webserver_enabled=true
--metrics_log_interval_ms=60000
```

```
--webserver_port=8050
#--webserver_doc_root=/data/kudu/www
```

2.12. 第十步：官网完整安装步骤

This script provides an overview of the procedure to build Kudu on a newly-installed RHEL or CentOS host, and can be used as the basis for an automated deployment scenario. It skips the steps marked **Optional** above.

```
#!/bin/bash

sudo yum -y install autoconf automake curl cyrus-sasl-devel cyrus-sasl-gssapi \
  cyrus-sasl-plain flex gcc gcc-c++ gdb git java-1.8.0-openjdk-devel \
  krb5-server krb5-workstation libtool make openssl-devel patch pkgconfig \
  redhat-lsb-core rsync unzip vim-common which
DTLS_RPM=rhscl-devtoolset-3-epel-6-x86_64-1-2.noarch.rpm
DTLS_RPM_URL=https://www.softwarecollections.org/repos/rhscl/devtoolset-3/epel-6-x86_64/noarch/${DTLS_RPM}
wget ${DTLS_RPM_URL} -O ${DTLS_RPM}
sudo yum install -y scl-utils ${DTLS_RPM}
sudo yum install -y devtoolset-3-toolchain
git clone https://github.com/apache/kudu
cd kudu
build-support/enable_devtoolset.sh thirdparty/build-if-necessary.sh
mkdir -p build/release
cd build/release
../../build-support/enable_devtoolset.sh \
  ../../thirdparty/installed/common/bin/cmake \
  -DCMAKE_BUILD_TYPE=release \
  ../..
make -j4
```

2.13. Kudu 集群启停

2.13.1. 启动集群

```
sudo service kudu-master start
sudo service kudu-tserver start
```

2.13.2. 查看进程状态

```
#通过systemctl status查看状态
sudo service kudu-master status
sudo service kudu-tserver status

#通过ps命令也可以查看进程
ps -aux | grep kudu

#通过查看端口监听也能判断进程状态
netstat -lntup

#设置为开机启动
sudo chkconfig kudu-master on
sudo chkconfig kudu-tserver on
```

2.13.3. 查看 web ui

2.13.3.1. master web ui

访问任意一个 master 内置的 webserver 即可，默认端口8051：http://bigdata02:8051

2.13.3.2. tserver webui

访问任意一个 tserver 内置的 webserver 即可，默认端口8050：http://bigdata02:8050

2.13.4. 关闭集群

```
sudo service kudu-tserver stop
sudo service kudu-master stop
```